

1 Resulting pending claims for examination:

2

3 1. A method for generating a dump file, the method comprising:

4 a. generating a dump file by gathering at least:

5 i. thread information for at least one running thread,

6 ii. context information for the thread,

7 iii. callstack information for the thread,

8 iv. process information for a process in which the thread

9 is running, and

10 v. information identifying a reason for generating the

11 dump file; and

12 b. storing the dump file to a storage medium.

13

14 2. The method as recited in Claim 1, further comprising determining when

15 to generate the dump file.

16

17 3. The method as recited in Claim 1, wherein generating the dump file

18 further includes gathering processor information about at least one

19 processor.

20

21 4. The method as recited in Claim 2, wherein determining when to

22 generate the dump file further includes determining that an exception

23 has occurred.

24

25

1 5. The method as recited in Claim 1, wherein the dump file does not
2 include data stored in global initialized memory.

3

4

5 6. The method as recited in Claim 1, wherein the dump file does not
6 include data stored in uninitialized memory.

7

8 7. The method as recited in Claim 1, wherein the dump file does not
9 include executable instructions used by a processor to execute a
10 program.

11

12 8. The method as recited in Claim 1, wherein the dump file is a kernel
13 minidump file associated with an operating system and the at least one
14 running thread is the single thread which encountered an exception.

15

16 9. The method as recited in Claim 8, wherein the callstack information
17 includes kernel stack information.

18

19 10. The method as recited in Claim 1, wherein the process information
20 identifies a process that initiated the thread.

21

22 11. The method as recited in Claim 1, further comprising:
23 allocating a buffer space in memory during an initialization
24 process, wherein the buffer space is suitable for storing the gathered
25 information; and

reserving space on the storage medium suitable for writing the contents of the buffer space.

12. The method as recited in Claim 11, wherein generating the dump file further includes initially storing the thread information, the context information, the callstack information, the process information, and the information identifying the reason for generating the dump file to the buffer space, and then copying the dump file from the buffer space to the storage medium as a minidump file.
13. The method as recited in Claim 12, further comprising upon re-initialization, after having stored the minidump file to the storage medium, accessing the minidump file on the storage medium and using at least a portion of the minidump file to further understand an exception that was at least one reason for generating the minidump file.
14. The method as recited in Claim 1, wherein the dump file is a user minidump file associated with at least one non-operating system program.
15. The method as recited in Claim 1, wherein generating the dump file further includes gathering callstack information for all running threads.
16. The method as recited in Claim 15, wherein the callstack information includes a user callstack.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

BL
AT

17. The method as recited in Claim 1, wherein generating the dump file further includes gathering processor context information for all running threads.
18. The method as recited in Claim 1, wherein generating the dump file further includes gathering a listing of loaded modules for a faulting application program.
19. The method as recited in Claim 1, wherein the dump file is a directory indexed file that uses relative virtual addresses (RVAs).
20. A computer-readable medium having computer-executable instructions for causing at least one processor to perform acts comprising:
gathering dump file information including at least thread information for at least one running thread, context information for the thread, callstack information for the thread, process information for the process in which the thread is running, and information identifying a reason for generating the dump file; and generating a dump file using the dump file information.
21. The computer-readable medium as recited in Claim 20, wherein generating the dump file further includes storing the dump file to a storage medium.

1

2 22. The computer-readable medium as recited in Claim 20, wherein

3 gathering the dump file information further includes gathering

4 processor information about at least one processor.

5

6 23. The computer-readable medium as recited in Claim 20, having further

7 computer-executable instructions for causing the at least one processor

8 to perform acts comprising determining when to generate the dump

9 file.

10

11 24. The computer-readable medium as recited in Claim 20, wherein the

12 dump file does not include data stored in global initialized memory.

13

14 25. The computer-readable medium as recited in Claim 20, wherein the

15 dump file does not include data stored in uninitialized memory.

16

17 26. The computer-readable medium as recited Claim 24 wherein the dump

18 file does not include executable instructions used by the at least one

19 processor to execute a program.

20

21 27. The computer-readable medium as recited in Claim 20, wherein the

22 dump file is a kernel minidump file associated with an operating

23 system and the at least one running thread is the single thread which

24 encountered an exception.

25

Bl
A

28. The computer-readable medium as recited in Claim 20, wherein the callstack information includes kernel stack information.

29. The computer-readable medium as recited in Claim 20, wherein the process information identifies a process that initiated the thread.

30. The computer-readable medium as recited in Claim 20, further comprising computer-executable instructions for causing the at least one processor to perform acts comprising:

allocating a buffer space in memory during an initialization process, wherein the buffer space is suitable for storing the dump file information; and

reserving space on a storage medium drive suitable for writing the contents of the buffer space.

31. The computer-readable medium as recited in Claim 30, wherein generating the dump file further includes initially storing the thread information, the context information, the callstack information, the process information, and the information identifying the reason for generating the dump file to the buffer space, and then copying the dump file from the buffer space to the storage medium as a minidump file.

BL
AT

- 32. The computer-readable medium as recited in Claim 31, further comprising computer-executable instructions for causing the at least one processor to perform acts comprising, upon re-initialization after having stored the minidump file to the storage medium, accessing the minidump file on the storage medium and using at least a portion of the minidump file to further understand an exception that was at least one reason for generating the minidump file.
- 33. The computer-readable medium as recited in Claim 20, wherein the dump file is a user minidump file associated with at least one non-operating system program.
- 34. The computer-readable medium as recited in Claim 20, wherein gathering the dump file information further includes gathering callstack information for all running threads.
- 35. The computer-readable medium as recited in Claim 34, wherein the callstack information includes a user callstack.
- 36. The computer-readable medium as recited in Claim 20, wherein gathering the dump file information further includes gathering processor context information for all running threads.

1 37. The computer-readable medium as recited in Claim 20, wherein
2 gathering the dump file information further includes gathering a listing
3 of all loaded modules for the faulting application program.

4

5 38. The computer-readable medium as recited in Claim 20, wherein the
6 dump file is a directory indexed file that uses relative virtual addresses
7 (RVAs).

8

9

10 39. An apparatus comprising;
11 memory;
12 a data storage drive configured to write data files to at least one data
13 storage medium; and
14 at least one processor operatively coupled to the memory and the data
15 storage drive and configured to:
16 a. generate a dump file by gathering in the memory at least:
17 i. thread information for at least one running thread,
18 ii. context information for the thread,
19 iii. callstack information for the thread,
20 iv. process information for the process in which the thread is
21 running, and
22 v. information identifying a reason for generating the dump
23 file; and
24 b. store the dump file to the storage medium.

25

1 40. The apparatus as recited in Claim 39, wherein the at least one processor
2 is further configured to determine when to generate the dump file.

3

4 41. The apparatus as recited in Claim 39, wherein the at least one processor
5 is further configured to gather processor information about the at least
6 one processor and include the processor information in the dump file.

7

8 42. The apparatus as recited in Claim 40, wherein the at least one processor
9 is further configured to determine when to generate the dump file
10 based on an exception.

11

12 43. The apparatus as recited in Claim 39, wherein the dump file does not
13 include data stored in global initialized memory.

14

15 44. The apparatus as recited in Claim 39, wherein the dump file does not
16 include data stored in uninitialized memory.

17

18 45. The apparatus as recited in Claim 39 wherein the dump file does not
19 include executable instructions used by the at least one processor to
20 execute a program.

21

22 46. The apparatus as recited in Claim 39, wherein the dump file is a kernel
23 minidump file associated with an operating system and the at least one
24 running thread is the single thread which encountered an exception.

25

1 47. The apparatus as recited in Claim 39, wherein the callstack information
2 includes kernel stack information.

3
4 48. The apparatus as recited in Claim 39, wherein the process information
5 identifies a process that initiated the thread.

6
7 49. The apparatus as recited in Claim 39, wherein the at least one processor
8 is further configured to:

9 allocate a buffer space in the memory during an initialization
10 process; and

11 reserve space on the storage medium drive suitable for writing the
12 contents of the buffer space.

13
14 50. The apparatus as recited in Claim 49, wherein the at least one processor
15 is further configured to:

16 generate the dump file by initially storing the thread information, the
17 context information, the callstack information, the process information,
18 and the information identifying the reason for generating the dump file
19 to the buffer space, and then copying the dump file from the buffer
20 space to the storage medium as a minidump file.

21
22 51. The apparatus as recited in Claim 50, wherein the at least one processor
23 is further configured to, upon re-initialization after having stored the
24 minidump file to the storage medium, access the minidump file on the
25 storage medium and use at least a portion of the minidump file to

1 further understand an exception that was at least one reason for
2 generating the minidump file.

3

4 B 52. The apparatus as recited in Claim 39, wherin the dump file is a user
5 minidump file associated with at least one non-operating system
6 program.

7 A 53. The apparatus as recited in Claim 39, wherein the at least one processor
8 is further configured to gather callstack information for all running
9 threads as part of the dump file.

10

11 54. The apparatus as recited in Claim 53, wherein the callstack information
12 includes a user callstack.

13

14 55. The apparatus as recited in Claim 39, wherein the at least one processor
15 is configured to gather processor context information for all running
16 threads as part of the dump file.

17

18 56. The apparatus as recited in Claim 39, wherein the at least one processor
19 is configured to gather a listing of all loaded modules for a faulting
20 application program as part of the dump file.

21

22 57. The apparatus as recited in Claim 39, wherein the dump file is a
23 directory indexed file that uses relative virtual addresses (RVAs).

24

25

1 67. The method as recited in Claim 1, further comprising providing the
2 dump file to at least one external device.

3

4 68. The method as recited in Claim 12, upon system re-initialization,
5 transferring the dump file from the storage medium to at least one
6 external device.

7

8 69. The method as recited in Claim 1, wherein generating the dump file
9 further includes gathering a list of loaded modules.

10

11 70. The computer-readable medium as recited in Claim 20, having further
12 computer-executable instructions for causing the at least one processor
13 to perform acts comprising providing the dump file to at least one
14 external device.

15

16 71. The computer-readable medium as recited in Claim 30, having further
17 computer-executable instructions for causing the at least one processor
18 to perform acts comprising, upon system re-initialization, transferring
19 the dump file from the storage medium to at least one external device.

20

21 72. The computer-readable medium as recited in Claim 20, wherein
22 gathering the dump file information further includes gathering a list of
23 loaded modules.

24

25

1 73. The apparatus as recited in Claim 39, wherein the at least one processor
2 is further configured to provide the dump file to at least one external
3 device.

4 74. The apparatus as recited in Claim 49, wherein the at least one processor
5 is further configured to, upon system re-initialization, transferring the
6 dump file from the storage medium to at least one external device.

7 75. The apparatus as recited in Claim 39, wherein the at least one processor
8 is further configured to gather a list of loaded modules as part of the
9 dump file

10 76. An application programming interface (API) method for use between a
11 first process and a second process operatively configured on at least
12 one processing unit in a computing device, the API method
13 comprising:
14 a. issuing, by the first process, a write dump file call having a
15 plurality of call parameters comprising a process handle, a
16 process identifier, a handle to a file where dump file information
17 is to be written, and a dump type identifier;
18 b. receiving, by the second process, the write dump file call and
19 parsing the call to retrieve the parameters; and
20 c. issuing, by the first process, a write dump file call
21 acknowledgment providing a true-false indication.

1 77. An application programming interface (API) method for use between a
2 first process and a second process operatively configured on at least
3 one processing unit in a computing device, the API method
4 comprising:

5 a. issuing, by the first process, a read dump file call having a
6 plurality of call parameters comprising a header of a dump file
7 and a data type identifier of data to read from a dump file;
8 b. receiving, by the second process, the read dump file call and
9 parsing the call to retrieve the parameters; and
10 c. issuing, by the first process, a read dump file call
11 acknowledgment providing a true-false indication and a plurality
12 of call return parameters comprising a pointer to a beginning of a
13 dump stream, and a stream size identifying the size of the dump
14 stream.

15

16

17

18

19

20

21

22

23

24

25

1 Claim amendments shown below:

2

3 1. (Once Amended) A method for generating a dump file, the method
4 comprising:

5 a. [determining when to generate a dump file; and
6 b.] generating a dump file by gathering at least:

7 i. thread information for at least one running thread,
8 ii. context information for the thread,
9 iii. callstack information for the thread,
10 iv. process information for a[the] process in which the
11 thread is running, and
12 v. information identifying a reason for generating the
13 dump file; and

14 b. storing the dump file to a storage medium.

15

16 2. (Once Amended) The method as recited in Claim 2, further comprising
17 determining when to generate [wherein generating] the dump file
18 [further includes storing the dump file to a storage medium].

19

20 5. (Once Amended) The method as recited in Claim 1[4], wherein
21 the dump file does not [further] include data stored in global initialized
22 [any significant portion of a dynamically allocated] memory.

1 6. (Once Amended) The method as recited in Claim 1[5] wherein
2 the dump file does not include [any portion of a global initialized or]
3 data stored in uninitialized memory.

4

5 7. (Once Amended) The method as recited in Claim 1[5] wherein
6 the dump file does not include [any portion of the] executable
7 instructions used by [the] a processor to execute [the] a program.

8

9 8. (Once Amended) The method as recited in Claim 1, wherein the
10 dump file is a kernel minidump file associated with an operating system
11 and the at least one running thread is the single thread which
12 encountered [the] an exception.

13

14 9. (Once Amended) The method as recited in Claim 8[1], wherein
15 the callstack information [is a] includes kernel stack information.

16

17 10. (Once Amended) The method as recited in Claim 1, wherein the
18 process information identifies [the] a process that initiated the thread.

19

20 11. (Once Amended) The method as recited in Claim 1, further
21 comprising:
22 [a.]allocating a buffer space in memory during an initialization process,
23 wherein the buffer space is suitable for storing the gathered
24 information; and

25

1 [b.] reserving space on [a] the storage medium [drive] suitable for
2 writing the contents of the buffer space.
3

4 12. (Once Amended) The method as recited in Claim 11, wherein[:
5 a.] generating the dump file further includes initially storing the thread
6 information, the context information, the callstack information, the
7 process information, and the information identifying the reason for
8 generating the dump file to the buffer space, and then copying the
9 dump file from the buffer space to the storage medium as a
10 minidump file[; and
11 b. upon system re-initialization, transferring the dump file from the
12 storage medium to at least one external computer].
13

14 13. (Once Amended) The method as recited in Claim 12, further
15 comprising upon re-initialization, after having stored the minidump file
16 to the storage medium, accessing the minidump file on the storage
17 medium and using at least a portion of the minidump file to further
18 understand an exception that was at least one reason for generating the
19 minidump file.
20

21 16. (Once Amended) The method as recited in Claim 15[1], wherein
22 the callstack information [is] includes a user callstack.
23
24
25

1 18. (Once Amended) The method as recited in Claim 1, wherein
2 generating the dump file further includes gathering a listing of [all]
3 loaded modules for [the] a faulting application program.

4

5 20. (Once Amended) A computer-readable medium having
6 computer-executable instructions for causing at least one processor to
7 perform[ing steps] acts comprising:

8 [a. determining when to generate a dump file; and
9 b. generating a dump file by] gathering dump file information
10 including at least[:

11 i.] thread information for at least one running thread,
12 [ii.] context information for the thread,
13 [iii.] callstack information for the thread,
14 [iv.] process information for the process in which the thread
15 is running, and
16 [v.] information identifying a reason for generating the
17 dump file; and

18 generating a dump file using the dump file information.

19

20 22. (Once Amended) The computer-readable medium as recited in Claim
21 20, wherein [generating] gathering the dump file information further
22 includes gathering processor information about at least one processor.

23

24 23. (Once Amended) The computer-readable medium as recited in Claim
25 20, [wherein] having further computer-executable instructions for

causing the at least one processor to perform acts comprising
determining when to generate the dump file [further includes
determining that an exception has occurred].

24. (Once Amended) The computer-readable medium as recited in Claim 20[23], wherein the dump file does not [further] include data stored in global initialized [any significant portion of a dynamically allocated] memory.

25. (Once Amended) The computer-readable medium as recited in Claim 20[24] wherein the dump file does not include [any portion of a global initialized or] data stored in uninitialized memory.

26. (Once Amended) The computer-readable medium as recited in Claim 24 wherein the dump file does not include [any portion of the] executable instructions used by the at least one processor to execute [the] a program.

27. (Once Amended) The computer-readable medium as recited in Claim 20, wherein the dump file is a kernel minidump file associated with an operating system and the at least one running thread is the single thread which encountered [the] an exception.

1 28. (Once Amended) The computer-readable medium as recited in Claim
2 20, wherein the callstack information [is a] includes kernel stack
3 information.

4
5 29. (Once Amended) The computer-readable medium as recited in Claim
6 20, wherein the process information identifies [the] a process that
7 initiated the thread.

8
9 30. (Once Amended) The computer-readable medium as recited in Claim
10 20, further comprising computer-executable instructions for causing the
11 at least one processor to perform[ing steps of] acts comprising:
12 allocating a buffer space in memory during an initialization process,
13 wherein the buffer space is suitable for storing the dump file information; and
14 reserving space on a storage medium drive suitable for writing the
15 contents of the buffer space.

16
17 31. (Once Amended) The computer-readable medium as recited in Claim
18 30, wherein generating the dump file further includes initially storing
19 the thread information, the context information, the callstack
20 information, the process information, and the information identifying
21 the reason for generating the dump file to the buffer space, and then
22 copying the dump file from the buffer space to the storage medium as a
23 minidump file[; and
24 upon system re-initialization, transferring the dump file from the
25 storage medium to at least one external different computer].

1

2 32. (Once Amended) The computer-readable medium as recited in Claim

3 31, further comprising computer-executable instructions for causing the

4 at least one processor to perform[ing steps of] acts comprising, upon

5 re-initialization after having stored the minidump file to the storage

6 medium, accessing the minidump file on the storage medium and using

7 at least a portion of the minidump file to further understand an

8 exception that was at least one reason for generating the minidump file.

9

10 34. (Once Amended) The computer-readable medium as recited in Claim

11 20, wherein [generating the dump file] gathering the dump file

12 information further includes gathering callstack information for all

13 running threads.

14

15 35. (Once Amended) The computer-readable medium as recited in Claim

16 34[20], wherein the callstack information [is] includes a user callstack.

17

18 36. (Once Amended) The computer-readable medium as recited in Claim

19 20, wherein [generating the dump file] gathering the dump file

20 information further includes gathering processor context information

21 for all running threads.

22

23 37. (Once Amended) The computer-readable medium as recited in Claim

24 20, wherein [generating the dump file] gathering the dump file

25

information further includes gathering a listing of all loaded modules for the faulting application program.

39. (Once Amended) An [arrangement] apparatus comprising;
memory[,];
a data storage drive configured to write data files to at least one data storage medium[,]; and
at least one processor operatively coupled to the memory and the data storage drive and configured to:
a. [determine when to generate a dump file; and
b.] generate a dump file by gathering in the memory at least:
i. thread information for at least one running thread,
ii. context information for the thread,
iii. callstack information for the thread,
iv. process information for the process in which the thread is running, and
v. information identifying a reason for generating the dump file; and
b. store the dump file to the storage medium.

40. (Once Amended) The [arrangement] apparatus as recited in Claim 39, wherein the at least one processor is further configured to determine when to generate[ing] the dump file [further includes storing the dump file to a storage medium].

1 41. (Once Amended) The [arrangement] apparatus as recited in Claim
2 39, wherein the at least one processor is further configured to
3 [generating the dump file further includes] gather[ing] processor
4 information about the at least one processor and include the processor
5 information in the dump file.

6

7 42. (Once Amended) The [arrangement] apparatus as recited in Claim
8 40[39], wherein the at least one processor is further configured to
9 determining when to generate the dump file [further includes
10 determining that] based on an exception [has occurred].

11

12 43. (Once Amended) The [arrangement] apparatus as recited in Claim
13 39[43], wherein the dump file does not [further] include data stored in
14 global initialized [any significant portion of a dynamically allocated]
15 memory.

16

17 44. (Once Amended) The [arrangement] apparatus as recited in Claim
18 39[43] wherein the dump file does not include [any portion of a global
19 initialized or] data stored in uninitialized memory.

20

21 45. (Once Amended) The [arrangement] apparatus as recited Claim
22 39[43] wherein the dump file does not include [any portion of the]
23 executable instructions used by the at least one processor to execute
24 [the] a program.

25

1 46. (Once Amended) The [arrangement] apparatus as recited in Claim
2 39, wherein the dump file is a kernel minidump file associated with an
3 operating system and the at least one running thread is the single thread
4 which encountered [the] an exception.

5
6 47. (Once Amended) The [arrangement] apparatus as recited in Claim
7 39, wherein the callstack information [is a] includes kernel stack
8 information.

9
10 48. (Once Amended) The [arrangement] apparatus as recited in Claim
11 39, wherein the process information identifies [the] a process that
12 initiated the thread.

13
14 49. (Once Amended) The [arrangement] apparatus as recited in Claim
15 39, wherein the at least one processor is further configured to [further
16 comprising computer-executable instructions for performing steps of]:
17 allocate[ing] a buffer space in the memory during an initialization
18 process; and
19 reserve[ing] space on [a] the storage medium drive suitable for
20 writing the contents of the buffer space.

21
22 50. (Once Amended) The [arrangement] apparatus as recited in Claim
23 49, wherein the at least one processor is further configured to:
24 generate[ing] the dump file [further includes] by initially storing the
25 thread information, the context information, the callstack information,

1 the process information, and the information identifying the reason for
2 generating the dump file to the buffer space, and then copying the
3 dump file from the buffer space to the storage medium as a minidump
4 file[; and

5 upon system re-initialization, transferring the dump file from the
6 storage medium to at least one external computer].

7

8 51. (Once Amended) The [arrangement] apparatus as recited in Claim
9 50, wherein the at least one processor is further [comprising computer-
10 executable instructions for performing steps of] configured to, upon re-
initialization after having stored the minidump file to the storage
11 medium, access[ing] the minidump file on the storage medium and
12 use[ing] at least a portion of the minidump file to further understand an
13 exception that was at least one reason for generating the minidump file.

14

15 52. (Once Amended) The [arrangement] apparatus as recited in Claim
16 39, wherein the dump file is a user minidump file associated with at
17 least one non-operating system program.

18

19 53. (Once Amended) The [arrangement] apparatus as recited in Claim
20 39, wherein the at least one processor is further configured to
21 [generating the dump file further includes] gather[ing] callstack
22 information for all running threads as part of the dump file.

1 54. (Once Amended) The [arrangement] apparatus as recited in Claim
2 53[39], wherein the callstack information [is] includes a user callstack.
3

4 55. (Once Amended) The [arrangement] apparatus as recited in Claim
5 39, wherein the at least one processor is configured to [generating the
6 dump file further includes] gather[ing] processor context information
7 for all running threads as part of the dump file.
8

9 56. (Once Amended) The [arrangement] apparatus as recited in Claim
10 39, wherein the at least one processor is configured to [generating the
11 dump file further includes] gather[ing] a listing of all loaded modules
12 for [the] a faulting application program as part of the dump file.
13

14 57. (Once Amended) The [arrangement] apparatus as recited in Claim
15 39, wherein the dump file is a directory indexed file that uses relative
16 virtual addresses (RVAs).
17

18 --67. The method as recited in Claim 1, further comprising providing the
19 dump file to at least one external device.
20

21 68. The method as recited in Claim 12, upon system re-initialization,
22 transferring the dump file from the storage medium to at least one
23 external device.
24

25

1 69. The method as recited in Claim 1, wherein generating the dump file
2 further includes gathering a list of loaded modules.

3

4 70. The computer-readable medium as recited in Claim 20, having further
5 computer-executable instructions for causing the at least one processor
6 to perform acts comprising providing the dump file to at least one
7 external device.

8

9 71. The computer-readable medium as recited in Claim 30, having further
10 computer-executable instructions for causing the at least one processor
11 to perform acts comprising, upon system re-initialization, transferring
12 the dump file from the storage medium to at least one external device.

13

14 72. The computer-readable medium as recited in Claim 20, wherein
15 gathering the dump file information further includes gathering a list of
16 loaded modules.

17

18 73. The apparatus as recited in Claim 39, wherein the at least one processor
19 is further configured to provide the dump file to at least one external
20 device.

21

22 74. The apparatus as recited in Claim 49, wherein the at least one processor
23 is further configured to, upon system re-initialization, transferring the
24 dump file from the storage medium to at least one external device.

25

75. The apparatus as recited in Claim 39, wherein the at least one processor is further configured to gather a list of loaded modules as part of the dump file.

76. An application programming interface (API) method for use between a first process and a second process operatively configured on at least one processing unit in a computing device, the API method comprising:

- issuing, by the first process, a write dump file call having a plurality of call parameters comprising a process handle, a process identifier, a handle to a file where dump file information is to be written, and a dump type identifier;
- receiving, by the second process, the write dump file call and parsing the call to retrieve the parameters; and
- issuing, by the first process, a write dump file call acknowledgment providing a true-false indication.

77. An application programming interface (API) method for use between a first process and a second process operatively configured on at least one processing unit in a computing device, the API method comprising:

- issuing, by the first process, a read dump file call having a plurality of call parameters comprising a header of a dump file and a data type identifier of data to read from a dump file;

- b. receiving, by the second process, the read dump file call and parsing the call to retrieve the parameters; and
- c. issuing, by the first process, a read dump file call acknowledgment providing a true-false indication and a plurality of call return parameters comprising a pointer to a beginning of a dump stream, and a stream size identifying the size of the dump stream. --